BITS Pilani, Pilani Campus
2nd Semester 2017-18
CS F211 Data Structures & Algorithms
Lab Test 1 - (18th Feb 2018)

Maximum Marks: 30 + 30 = 60 **Duration: 160 + 10 = 170 min**

Instructions

- In this lab test you have to solve two problems given in two different files. Total time for solving the problems is 160 minutes with at most 10 additional minutes for uploading the solution on the portal. Portal closing timer is shown at top-right corner of your page.
- You may make multiple submissions on the online portal, but only the latest submission shall be considered for evaluation.
- For each of the given two problems, upload all relevant source code (*.c) and header (*.h) files as a single submission. Do not upload any other file.
- If your program is having a "*compilation error*" or "*segmentation fault*" or other similar errors, we shall **evaluate your code for 50% of the marks only**. So, try to keep your code as clean as possible. Also note that **commented code** will **NOT** be considered for evaluation.
- Each problem is associated with a zip file which contains a "**main.c**" file and a few additional files:
  - DO **NOT** make any changes in the file - "**main.c**" in either of the problems.
  - Write your code in the provided additional files only. Do NOT rename any file, or create a new file.
  - Please note that each function may be tested separately for various test cases. So, don't make changes to the signatures and return types of the pre-defined functions.

Problem 2 of 2

[Expected Time: 75 minutes]

Your input file contains the number of records in its first line and one student record per subsequent line. Each record has two fields separated by one space character: <`name`> a string of max size 10 and <`marks`> an **int** value. You can use the files in **q2.zip** and build your code over them. A sample input file – Input.txt is also provided in the zip. **DO NOT** make any changes to the **signatures** and **return types** of the predefined functions in these files. You may also add more functions as required.

a. Implement a function – `readData`(), that takes in a string containing the file name, reads the file into an array of integer values containing the marks only. This function should also store the size of the array in a global variable called size (declared in "qsort.h"). Don't change the signature and return type of this function. **[5M]**

b. Implement a θ(n)-time - "`extractKeys`()" function that takes an array `Ls` of `n` integer records, the size `n`, and finds all the keys in it and returns them in a sorted array named Keys along with its size. Keys should not contain any duplicates. It is known that each record in `Ls` will have a key in the range `loKey..hiKey` (taken as inputs to the function) and all values in this range need not occur as a key. **[5M]**

c. Implement a θ(n)-time locality-aware partitioning function – `part2Loc`() that takes an array `Ls` of `n` integers, the lower index `lo` and higher index `hi` of `Ls`, and an integer pivot (`key`) as arguments and partitions `Ls` into two sub-lists based on the pivot. **[10M]**

d. Use your `part2Loc()` function and the `extractKeys()` to implement an iterative QuickSort algorithm – `quickSortKnownKeyRange()` for the special case where the number of unique keys is much less than the number of records. This function takes the array `Ls` of `n` integers, its `size`, and `loKey` & `hiKey` values as defined in Problem 2b. This QuickSort algorithm should call your partition procedure no more than K-1 times and the total time complexity of sorting should be no more than θ(K*N) where N is number of records and K is the size of the Keys array as returned by extractKeys(). **[10M]**